Entanglement on mixed stabiliser states: Normal Forms and Reduction Procedures

Koenraad M.R. Audenaert and Martin B. Plenio
Blackett Laboratory, Imperial College London, Prince Consort Road, London SW7 2BW, United Kingdom and
Institute for Mathematical Sciences, Imperial College London, Exhibition Road, London SW7 2BW, United Kingdom

(Dated: February 1, 2008, 9:44)

The stabiliser formalism allows the efficient description of a sizeable class of pure as well as mixed quantum states of n-qubit systems. That same formalism has important applications in the field of quantum error correcting codes, where mixed stabiliser states correspond to projectors on subspaces associated with stabiliser codes.

In this paper, we derive efficient reduction procedures to obtain various useful normal forms for stabiliser states. We explicitly prove that these procedures will always converge to the correct result and that these procedures are efficient in that they only require a polynomial number of operations on the generators of the stabilisers.

On one hand, we obtain two single-party normal forms. The first, the row-reduced echelon form, is obtained using only permutations and multiplications of generators. This form is useful to calculate partial traces of stabiliser states. The second is the fully reduced form, where the reduction procedure invokes single-qubit operations and CNOT operations as well. This normal form allows for the efficient calculation of the overlap between two stabiliser states, as well as of the Uhlmann fidelity between them, and their Bures distance.

On the other hand, we also find a reduction procedure of bipartite stabiliser states, where the operations involved are restricted to be local ones. The two-party normal form thus obtained lies bare a very simple bipartite entanglement structure of stabiliser states. To wit, we prove that every bipartite mixed stabiliser state is locally equivalent to a direct product of a number of maximally entangled states and, potentially, a separable state. As a consequence, using this normal form we can efficiently calculate every reasonable bipartite entanglement measure of mixed stabiliser states.

PACS numbers: 03.67.-a,03.65.Yz

I. INTRODUCTION

The exploration of the properties of quantum entanglement is one of the main branches of quantum information theory [1, 2, 3]. While a reasonably detailed understanding of two-qubit entanglement has been achieved, the entanglement properties of higher-dimensional or multi-particle systems remain largely unexplored, with only isolated results [4]. This is largely due to the complexity involved in these investigations, which in turn originates from the tensor product structure of the multi-particle state space. This structure leads to an exponential growth in the number of parameters that are required for the description of the state. The same problem arises when one attempts to consider the time-evolution of a many-body quantum system or, say, of a quantum computation. Generally, a significant part of the Hilbert space and consequently an exponential number of parameters are required to describe the quantum system at all times.

One way of approaching this situation is to impose constraints on the set of states and/or the set of operations that one is interested in without curtailing the variety of possible qualitative entanglement structures too much. In this context an interesting class of states that arises is that of stabiliser states [5, 6, 7, 8, 9, 10] which, via the concept of graph states [11, 12, 13, 14, 15, 16, 17, 18] have some connection with graph theory. The feature of these states that allows for a more detailed study of their entanglement properties is the fact that an n-particle stabiliser state is determined as the joint unique eigenvector with eigenvalue $+1$ of a set of only n tensor products of Pauli operators. This results in a very compact description of the quantum state requiring only of order $O(n^2)$ parameters and therefore provides hope that a more detailed understanding of their entanglement structure can be obtained. Despite this simplification, stabiliser states still exhibit multi-particle entanglement and permit, for example, the violation of local realism [15].

The stabiliser formalism not only allows for the efficient description of a certain type of quantum states, but also permits the efficient simulation of a restricted, but nevertheless interesting, class of time evolutions, namely those that respect the tensor product structure of the Pauli operators [19, 20, 21, 22]. Again, these simulations can be performed in polynomial time in the number of qubits, in stark contrast to the simulation of a general time evolution of an n-qubit system, which requires an amount of resources that is exponential in n.

The stabiliser formalism uniquely specifies the quantum state of an n-qubit system employing only polynomial resources. This alone, however, is not sufficient. It is also important to be able to derive *all* relevant physical quantities, especially those relating to entanglement, directly from the stabiliser formalism. Indeed, having first to deduce the state explicitly and then computing the property from the state would generally involve an undesirable exponential overhead in resources. While one can expect a direct approach to be possible in principle, it is evident that detailed and explicit presentations of algorithms to achieve these tasks in a systematic way and whose convergence is proven are of significant interest. In the context of entanglement properties some effort has recently been expended in this direction in [23], where, employing sophisticated tools from group theory, the existence of a useful entanglement measure for multi-particle stabiliser states was demonstrated.

The present work progresses further in a similar direction. Employing elementary tools we present a number of normal forms for pure *and* mixed stabiliser states, together with explicit and detailed descriptions of algorithms (including proofs of convergence) that allow the reduction to these normal forms. In turn, these normal forms then permit us to compute *any* entanglement measure, overlaps between stabiliser states and various other quantities. Detailed descriptions of the algorithms are provided that should make it straightforward to implement these algorithms in any programming environment and we are able to provide a (β-tested) suite of MatLab programs on request.

This suite can then form the basis for more detailed studies and further applications of the stabiliser formalism to a whole range of physical questions (see also [20]). This will be reported on in a forthcoming publication.

The stabiliser formalism also plays a central role in the field of quantum error correcting codes. Mixed stabiliser states (defined in Section II) are in one-to-one correspondence with projectors on subspaces associated with stabiliser codes [5]. Although our normal forms and reduction procedures have been designed with applications to entanglement theory in mind, they might have a bearing on stabiliser codes as well.

The present paper is organized as follows: In Section II the basic notations and conventions we use are introduced while Section III describes the elementary operations that will form the basis of all reduction procedures.

The single-party normal forms are the topic of Sections IV, V and VI. Section IV deals with the so-called row-reduced echelon form (algorithm RREF), the reduction to which is based on row operations only. It allows to check independence of any (putative) set of generators and to calculate partial traces (algorithm PTRACE). In Section V we describe the full reduction procedure (algorithm CNF1) to single-party normal form, using row and column (qubit) operations. In Section VI we present an algorithm (algorithm OVERLAP) that is based on the full reduction and allows to calculate overlaps between stabiliser states, Uhlmann fidelity, and Bures distance.

In Section VII we turn to the bipartite case, where we prove that the bipartite entanglement structure of stabiliser states is remarkably simple. To wit, we show that mixed bipartite stabiliser states are locally equivalent to a tensor product of a certain number of maximally entangled 2-qubit states and, potentially, a fully sparable mixed state. We present an algorithm (algorithm CNFP) to obtain the number of these maximally entangled pairs, allowing for the calculation of any reasonable bipartite entanglement measure.

We conclude the description of our findings in Section VIII.

II. NOTATIONS AND CONVENTIONS

A *stabiliser operator* on N qubits is a tensor product of operators taken from the set of Pauli operators

$$X := \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix},\ Y := \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix},\ Z := \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \quad (1)$$

and the identity $\mathbb{1}$. An example for $N = 3$ would be the operator $g = X \otimes \mathbb{1} \otimes Z$. A set $G = \{g_1, \ldots, g_K\}$ of K mutually commuting stabiliser operators that are independent, i.e. $\prod_{i=1}^K g_i^{s_i} = \mathbb{1}$ exactly if all s_i are even, are called the *generator set* for the *stabiliser group* S. This stabiliser group S then consists of all distinct products of operators from the generator set.

For $K = N$ a generator set G uniquely determines a single state $|\psi\rangle$ that satisfies $g_k|\psi\rangle = |\psi\rangle$ for all $k = 1, \ldots, N$. Any state for which such a generator set exists is called *stabiliser state*. Such a state has trivially the property that $g_k|\psi\rangle\langle\psi| = |\psi\rangle\langle\psi|$ for all k so that

$$|\psi\rangle\langle\psi| = \frac{1}{2^N}\sum_{g\in S} g. \quad (2)$$

This formula depends on the complete set of stabilisers, and is, therefore, not very practical. The following formula expresses the state in terms of a generator set [8]

$$|\psi\rangle\langle\psi| = \prod_{k=1}^N \frac{\mathbb{1} + g_k}{2}. \quad (3)$$

The procedure presented in Section IV yields as a side result an elementary proof of this statement.

Considering two parties A and B, the reduced density matrix of the stabiliser state can be computed as

$$\rho_A = \mathrm{Tr}_B\,\rho = \frac{1}{2^N}\sum_{g\in S} \mathrm{Tr}_B\, g. \quad (4)$$

This obviously means that only operators g contribute that have identity operators acting on all qubits belonging to B. Needless to say, computing ρ_A from ρ directly is hopelessly inefficient; as there are 2^N different g, this task requires an exponential number of operations in general.

It turns out, however, that there is a class of mixed states that can also be characterised employing stabilisers. We will call these *mixed stabiliser states*, and they contain the pure stabiliser states as a subset. The important feature of this class is that the reduced density matrix of a mixed stabiliser state is again a mixed stabiliser state. Furthermore, as we will show below in Section IV C, the stabiliser group of a reduction can be efficiently calculated directly from the original stabiliser group, without calculating the state and its reduction explicitly.

To characterise mixed stabiliser states, one simply considers sets G that are linearly dependent. As a consequence, by multiplying stabilisers, one can achieve that some of them become identical to $\mathbb{1}$ and only K linearly independent ones remain. Then the common eigenspace of these K operators will have a dimension larger than 1. As in eq. (3) one immediately deduces that the density operator is again just the projector onto this eigenspace, rescaled to trace 1:

$$\begin{aligned} \mathcal{P} &= \prod_{k=1}^K \frac{\mathbb{1} + g_k}{2} \\ \rho &= \frac{1}{2^{N-K}}\mathcal{P}. \end{aligned} \quad (5)$$

Given that $\mathcal{P}$ is a projector onto a subspace of dimension 2^{N-K}, the entropy of ρ is simply $N - K$. In analogy with matrix analysis terminology, we will call K the *rank* of the stabiliser group. Stabiliser groups with $K = N$ will be called *full-rank*, and stabiliser groups with $K < N$ *rank-deficient*.

Note: In case of a rank-deficient stabiliser group one has to distinguish between *stabiliser* states and *stabilised* states. The stabiliser state is the one given by (5), and the stabiliser formalism allows to study its properties in an efficient way. On the other hand, there are many states that are stabilised by that same stabiliser group, but in general they are not stabiliser states. Indeed, most of these stabilised states cannot be described as "the unique state stabilised by a full-rank stabiliser group", and hence, the stabiliser formalism cannot be used to study their properties via that group. For example, any state is stabilised by the (singleton) stabiliser group $\{\mathbb{1}\}$, but only the maximally mixed state $\mathbb{1}/2$ is the stabiliser state corresponding to that group.

For the following the aim will be to derive basic entanglement properties such as the entropy of entanglement or the logarithmic negativity for stabiliser states, pure or mixed, directly from their generating set. To this end it will be useful to find a normal form for the generator set that reveals the relevant entanglement structure.

We now introduce the concept of *stabiliser array*, which is quite simply a rectangular array of K rows and N columns, where the elements are Pauli matrices or the Identity matrix. Specifically, the element in the k-th row and n-th column of the stabiliser array corresponding to a generator set $G = \{g_1, \ldots, g_K\}$ on N qubits is the n-th tensor factor (corresponding to qubit n) of the k-th generator g_k. For some applications it will also be necessary to deal with the generator phase factors. While in general these phase factors can assume the values ± 1 and $\pm i$, for the purpose of describing stabiliser states only the values ± 1 make sense (since states are Hermitian). We will store these phase factors in a K-dimensional vector s, where s_k is the phase factor of generator g_k.

The purpose of the various normal forms that will be presented in this paper are to structure the set of stabiliser states into certain equivalence classes. They are similar in spirit to the normal forms that have been devised for matrices. For example, the row-reduced echelon form, which exhibits the rank of a matrix, has a counterpart for stabiliser arrays. Despite this similarity, the normal forms presented here are of an entirely different nature. The rank of a stabiliser array, which we will introduce in Section IV, is akin to the rank of a matrix in that it equals the number of independent generators in a generator set, but there the similarity stops. In linear algebra one defines both row and column rank of a matrix and one proves that these two ranks are actually equal. For stabiliser arrays, one cannot even give a meaningful definition of column rank.

These differences ultimately boil down to the fact that a stabiliser array is not really a matrix. The two foremost reasons are that its elements are not numbers but elements of the Pauli group, and second, that matrices represent linear operations in linear spaces, while stabiliser arrays represent sets (namely, sets of generators). As a consequence, while operations like matrix transpose, matrix multiplication, addition, and inverse make perfect sense for matrices, they are utterly meaningless for stabiliser arrays. The allowed operations on stabiliser arrays are thus much more restricted than in the matrix case. For example, the only row operations that make sense for stabiliser arrays are row interchange and elementwise row multiplication (which is based on the Pauli group multiplication law). This will be discussed in more detail in the following Section. This explains the need for entirely new reduction procedures for stabiliser arrays.

	$\mathbb{1}$	X	Y	Z
$\mathbb{1}$	$\mathbb{1}$	X	Y	Z
X	X	$\mathbb{1}$	iZ	$-iY$
Y	Y	$-iZ$	$\mathbb{1}$	iX
Z	Z	iY	$-iX$	$\mathbb{1}$

TABLE I: Multiplication table for Pauli operators; shown is $\sigma_{\text{row}}.\sigma_{\text{col}}$.

III. ELEMENTARY OPERATIONS

In this Section, we describe the allowed elementary operations that transform a stabiliser array and which we will use to reduce an array to its normal forms. As in the matrix case, these operations come in two kinds. The first kind are the row operations. It is important to realise that row operations will not alter the stabiliser state at all, but only alter the generator set it is represented by. These are the *row transposition*, which interchanges (transposes) two rows in the stabiliser array, and the *row multiplication*, which multiplies one row with another one. The latter operation changes the generators of the stabiliser group, but not the group itself and hence not the stabiliser state either. We will use the phrase "multiply row k with row l" to mean "multiply rows k and l elementwise and set row l to the product obtained." The multiplication table for Pauli operators is shown in Table I.

The second kind of operations are the column operations, which may alter the state. The column operations we will use are a certain class of single-qubit operations, transposing two columns, and the CNOT operation between two qubits. As single-qubit operations we take those that act on one given column of the stabiliser operators by permuting the Pauli operators (in the given column) among themselves. These operations can be constructed from combinations of Hadamard gates (H) and $\pi/4$ gates (P) (see Table II). Note that odd permutations must involve a sign change in one of the Pauli operators in order to correspond to a unitary operation. The particular sign changes of Table II have been chosen to make the unitaries implementing the odd permutations involutory (apart from a global phase). That is, $UU = \exp(i\phi)\mathbb{1}$. Note also that the second and third permutation in the Table are each other's inverse.

For the bipartite normal form described in Section VII we will need to divide the qubits into two parties and only allow operations that are local to those parties. Transposing

X	Y	Z	Unitary
X	Y	Z	$\mathbb{1}$
Z	X	Y	PH
Y	Z	X	$HP^\dagger$
$-X$	Z	Y	$PHP^\dagger$
Y	X	$-Z$	$HPPHP^\dagger$
Z	$-Y$	X	H

TABLE II: Truth table for the single-qubit operations employed by the CNF algorithm. Any permutation of the set of Pauli operators can be achieved.

C	T	C'	T'
$\mathbb{1}$	$\mathbb{1}$	$\mathbb{1}$	$\mathbb{1}$
$\mathbb{1}$	X	$\mathbb{1}$	X
$\mathbb{1}$	Y	Z	Y
$\mathbb{1}$	Z	Z	Z
X	$\mathbb{1}$	X	X
X	X	X	$\mathbb{1}$
X	Y	Y	Z
X	Z	$-Y$	Y

C	T	C'	T'
Y	$\mathbb{1}$	Y	X
Y	X	Y	$\mathbb{1}$
Y	Y	$-X$	Z
Y	Z	X	Y
Z	$\mathbb{1}$	Z	$\mathbb{1}$
Z	X	Z	X
Z	Y	$\mathbb{1}$	Y
Z	Z	$\mathbb{1}$	Z

TABLE III: Truth table for the CNOT gate employed by the CNF algorithm. C and T refer to control and target qubit, respectively. The primed columns give the values after the operation.

two columns in the bipartite case is only allowed when both columns (qubits) belong to the same party. Otherwise this would be a non-local operation, which would very likely affect the amount of entanglement.

The CNOT gate between two qubits, one being the control qubit and one the target qubit, operates on the two corresponding columns of the stabiliser array. In the case of a bipartite CNF, we must again ensure that both qubits belong to the same party. The truth table for the CNOT is given in table III. Note that the column pertaining to the control qubit is modified too; this is a peculiarity of the description of states by stabilisers.

IV. ROW-REDUCED ECHELON FORM

While the Clifford Normal Form (CNF) of a stabiliser array will be obtained below via application of both elementary row and column operations, it is possible to obtain a normal form using elementary row operations only. Due to its similarity to the matrix case, we will call this normal form the *Row-Reduced Echelon Form* (RREF). The benefits of the RREF are that it is very easy to obtain, the stabiliser state represented by the stabiliser array is not changed, and it is applicable to states on any number of parties. Furthermore, as we shall see below, it is an efficient way to eliminate linearly dependent rows from the stabiliser array.

The general structure of the RREF is most easily described in a recursive fashion. There are three cases:

$$\left(\begin{array}{c|c} \mathbb{1} & \\ \vdots & \text{RREF}' \\ \mathbb{1} & \end{array}\right), \quad \left(\begin{array}{c|c} \sigma & *\ldots* \\ \hline \mathbb{1} & \\ \vdots & \text{RREF}' \\ \mathbb{1} & \end{array}\right) \quad \text{and} \quad \left(\begin{array}{c|c} \sigma_1 & *\ldots* \\ \sigma_2 & *\ldots* \\ \hline \mathbb{1} & \\ \vdots & \text{RREF}' \\ \mathbb{1} & \end{array}\right).$$

The symbols '$\vdots$' and '$\ldots$' denote a number of repeated rows and columns. This number may be zero. The symbol RREF$'$ denotes a (possibly empty) sub-array that is also in RREF form. The symbol $*$ denotes either a Pauli operator or an identity $\mathbb{1}$. Furthermore, σ, σ_1 and σ_2 are Pauli operators, and σ_1 and σ_2 anticommute. We will refer to the operators in these positions as *column leaders* of their column, and *row leaders* of their row.

The RREF algorithm works by applying a sequence of elementary row operations to the stabiliser array. At every step of the algorithm it is determined which elementary operation to apply based on the values contained in a certain contiguous subarray of the full array. At every step this subarray, which we will call the *active region*, either stays the same or decreases in size. The algorithm terminates when the size of the active region has decreased to zero. Note that the elementary operations operate on the full stabiliser array and not just on the active region.

Let K, N be the number of rows (generators) and columns (qubits) of the stabiliser array, respectively. The variable K_U contains the index of the first row in the active region, and N_L the index of its first column. The active region thus consists of the array elements (i, j) for $K_U \le i \le K$ and $N_L \le j \le N$. Initially, the active region comprises the full stabiliser array, hence $K_U = 1$ and $N_L = 1$.

In this and subsequent sections, the phase factors $\exp(i\phi_k)$ of the various generators will not be mentioned explicitly. They are best maintained under the form of a single additional column in the stabiliser array, which is modified by row permutations and the elementary operations of tables I, II and III.

A. Algorithm RREF

1. Count the number of different Pauli operators (X, Y and Z) in the first column (N_L) of the active region, i.e. restricting attention only to rows K_U up to K.

2. Three cases can be considered:

   (a) *There are no Pauli operators in column N_L.*

      i. Increase N_L by 1.

   (b) *There is only 1 kind of Pauli operator.*
   Let k be the first row in the active region where column N_L contains a Pauli operator.

      i. Make row k the top row of the active region by transposing, if necessary, row k with row K_U.

Initial stabiliser array:

$$\begin{pmatrix} \sigma_1 \\ \sigma_2 \\ . \end{pmatrix}$$

Depending on the content of row 3, do the following:

$\mathbb{1}$: Do nothing.

σ_1: Multiply row 1 with row 3.

σ_2: Multiply row 2 with row 3.

σ_3: Multiply row 1 with row 3, and then row 2 with row 3.

TABLE IV: Required operations to eliminate any Pauli operator from row 3 of the stabiliser array shown above. The operators σ_1, σ_2, and σ_3 are a permutation of X, Y and Z.

ii. Multiply row K_U with all other rows in the active region that have the same Pauli in column N_L.

iii. Increase K_U and N_L by 1.

(c) *There are at least 2 different kinds of Pauli operators.*
Let k_1 be the first row in the active region where column N_L contains a Pauli operator, and k_2 be the first row in the active region where column N_L contains a different Pauli operator.

i. Make row k_1 the top row of the active region by transposing, if necessary, row k_1 with row K_U.

ii. Make row k_2 the second row of the active region by transposing, if necessary, row k_2 with row $K_U + 1$.

iii. Multiply every other row in the active region with either row K_U, row $K_U + 1$, both rows or none, depending on the element in column N_L (see Table IV).

3. If the active region still has non-zero size ($N_L \leq N$ and $K_U \leq K$), continue with step 1, else terminate.

B. Checking independence of a set of generators

The easiest way to check independence of a set of generators is to compute the RREF of the stabiliser array. This fact is one other property the stabiliser RREF and the matrix RREF have in common. Dependencies between generators will show up as RREF rows containing only $\mathbb{1}$ operators. Removing these all-$\mathbb{1}$ rows leaves an independent set of generators.

Proof. From the form of the RREF one observes that there cannot be more than two rows with the same number of leading $\mathbb{1}$ operators, and if there are two such, they have a different row leader. Consider a subset of generators g_k in the RREF, having n_k leading $\mathbb{1}$ operators, and having σ_k as row leaders. Let the generators be sorted according to n_k in ascending order. When multiplying two rows that satisfy $n_2 \geq n_1$, the number of leading $\mathbb{1}$ operators in the product is n_1 and the row leader is either σ_1 (if $n_2 > n_1$) or $\sigma_1\sigma_2$ (if $n_2 = n_1$), which is different from either σ_1 and σ_2. In both cases this shows that the product cannot occur as another generator in the RREF. This proves that it is not possible to write one RREF generator as a product of other RREF generators. □

C. Partial Trace of a Stabiliser State

A useful and important operation is the partial trace. The RREF algorithm is the central part in the following efficient partial trace algorithm:

Algorithm PTRACE

1. By column permutations bring the columns of the qubits to be traced out in first position.
2. Bring those columns to RREF.
3. Remove the rows containing the column leader(s).
4. Finally, remove those columns themselves.

Proof. To prove that this algorithm indeed calculates the partial trace, consider again the three cases for the RREF:

$$\left(\begin{array}{c|c} \mathbb{1} & \\ \vdots & \text{RREF}' \\ \mathbb{1} & \end{array}\right), \quad \left(\begin{array}{c|c} \sigma & *\ldots* \\ \hline \mathbb{1} & \\ \vdots & \text{RREF}' \\ \mathbb{1} & \end{array}\right) \text{ and } \left(\begin{array}{c|c} \sigma_1 & *\ldots* \\ \sigma_2 & *\ldots* \\ \hline \mathbb{1} & \\ \vdots & \text{RREF}' \\ \mathbb{1} & \end{array}\right).$$

We have to show that the state described by RREF$'$, say ρ', is the state obtained from the original stabiliser state ρ by tracing out the qubit pertaining to column 1. Denote the sequences of $*$ operators by g, g_1 and g_2, respectively.

Using eq. (5), it is easy to see that, in the first case,

$$\rho = \frac{\mathbb{1}}{2} \otimes \rho',$$

and tracing out qubit 1 yields

$$\text{Tr}_1\,\rho = \rho'.$$

In the second case,

$$\begin{aligned} \rho &= \frac{\mathbb{1} \otimes \mathbb{1} + \sigma \otimes g}{2} (\mathbb{1} \otimes \rho') \\ &= \frac{1}{2}(\mathbb{1} \otimes \rho' + \sigma \otimes g\rho'), \end{aligned}$$

and again, as Pauli operators have trace 0,

$$\text{Tr}_1\,\rho = \rho'.$$

In the third and final case,

$$
\begin{aligned}
\rho &= \frac{\mathbb{1}\otimes\mathbb{1}+\sigma_1\otimes g_1}{2}\,\frac{\mathbb{1}\otimes\mathbb{1}+\sigma_2\otimes g_2}{2}\,(\mathbb{1}\otimes 2\rho') \\
&= \frac{1}{2}(\mathbb{1}\otimes\rho' + \sigma_1\otimes g_1\rho' + \sigma_2\otimes g_2\rho' \\
&\quad +\sigma_1\sigma_2\otimes g_1 g_2\rho'),
\end{aligned}
$$

resulting yet again in

$$\mathrm{Tr}_1\,\rho = \rho'.$$

□

V. SINGLE-PARTY NORMAL FORM

The CNF algorithm works by applying a sequence of elementary operations to the stabiliser array. At every step of the algorithm it is determined which elementary operation to apply based on the values contained in a certain contiguous subarray of the full array. At every step this subarray, which we will call the *active region*, either stays the same or decreases in size. The algorithm terminates when the size of the active region has decreased to zero. Note that the elementary operations operate on the full stabiliser array and not just on the active region.

Let K, N be the number of rows (generators) and columns (qubits) of the stabiliser array, respectively. The variables K_U and K_L contain the indices of the first (uppermost) and last (lowermost) row in the active region, and N_L and N_R the indices of its first (leftmost) and last (rightmost) column. The active region thus consists of the array elements (i, j) for $K_U \leq i \leq K_L$ and $N_L \leq j \leq N_R$. Initially, the active region comprises the full stabiliser array, hence $K_U = 1$, $K_L = K$, $N_L = 1$ and $N_R = N$. We will prove below that after every iteration of the algorithm the stabiliser array has the block structure

$$
\left(\begin{array}{cccc|ccc|ccc}
X & \mathbb{1} & \dots & \mathbb{1} & \mathbb{1} & \dots & \mathbb{1} & \mathbb{1} & \dots & \mathbb{1} \\
\mathbb{1} & X & \dots & \mathbb{1} & \mathbb{1} & \dots & \mathbb{1} & \mathbb{1} & \dots & \mathbb{1} \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\mathbb{1} & \mathbb{1} & \dots & X & \mathbb{1} & \dots & \mathbb{1} & \mathbb{1} & \dots & \mathbb{1} \\
\hline
\mathbb{1} & \mathbb{1} & \dots & \mathbb{1} & * & \dots & * & \mathbb{1} & \dots & \mathbb{1} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\mathbb{1} & \mathbb{1} & \dots & \mathbb{1} & * & \dots & * & \mathbb{1} & \dots & \mathbb{1}
\end{array}\right).
$$

The block containing the asterisks is the active region and has not yet been brought to normal form. The columns on the left of the active region correspond to qubits that are in an eigenstate of the X operator, the columns on its right correspond to qubits that are in a totally mixed state. The final form, after completion of the algorithm, is

$$
\left(\begin{array}{cccc|ccc}
X & \mathbb{1} & \dots & \mathbb{1} & \mathbb{1} & \dots & \mathbb{1} \\
\mathbb{1} & X & \dots & \mathbb{1} & \mathbb{1} & \dots & \mathbb{1} \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\
\mathbb{1} & \mathbb{1} & \dots & X & \mathbb{1} & \dots & \mathbb{1} \\
\hline
\mathbb{1} & \mathbb{1} & \dots & \mathbb{1} & \mathbb{1} & \dots & \mathbb{1} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\mathbb{1} & \mathbb{1} & \dots & \mathbb{1} & \mathbb{1} & \dots & \mathbb{1}
\end{array}\right).
$$

Here we have left open the possibility that the rows of the initial stabiliser array might not be independent.

A. Algorithm CNF1

1. Count the number of different Pauli operators (X, Y and Z) in the first column (N_L) of the active region, i.e. restricting attention only to rows K_U up to K_L.
2. Three cases can be considered:
   (a) *There are no Pauli operators in column N_L.*
      i. If necessary, transpose column N_L with column N_R.
      ii. Decrease N_R by 1.
   (b) *There is only 1 kind of Pauli operator.*
   Let k be the first row in the active region where column N_L contains a Pauli operator.
      i. Make row k the top row of the active region by transposing, if necessary, row k with row K_U.
      ii. Apply whatever single-qubit operation on column N_L that brings that Pauli operator to an X.
      iii. Multiply row K_U with all other rows in the active region that have an X in column N_L.
      iv. Consider the elements of the first row of the active region (row K_U). To each of the columns beyond the first one that contains in the first row a Pauli different from X, apply a single-qubit operation to turn it into an X.
      v. To each of these columns, which now have an X in the first row, successively apply a CNOT operation with control column N_L.
      vi. Increase K_U and N_L by 1.
   (c) *There are at least 2 different kinds of Pauli operators.*
   Let k_1 be the first row in the active region where column N_L contains a Pauli operator, and k_2 be the first row in the active region where column N_L contains a different Pauli operator.
      i. Make row k_1 the top row of the active region by transposing, if necessary, row k_1 with row K_U.

ii. Make row k_2 the second row of the active region by transposing, if necessary, row k_2 with row $K_U + 1$.

iii. Bring the element on row K_U to an X and the element on row $K_U + 1$ to a Z by applying, if necessary, a single-qubit operation on column N_L.

iv. Consider the first two rows of the active region (rows K_U and $K_U + 1$). Find the first column beyond column N_L, say column l, that contains an anticommuting pair on those rows (i.e. two non-identical Pauli operators).

v. Bring the anticommuting pair to an (X, Y) pair by applying, if necessary, a single-qubit operation to that column.

vi. Apply a CNOT operation to that column, with column N_L as control.

vii. The extent of the active region is not changed in this case.

3. If the active region still has non-zero size ($N_L \leq N_R$ and $K_U \leq K_L$), continue with step 1, else terminate.

B. Proof of correctness of algorithm CNF1

We will now show that algorithm CNF1 indeed brings any stabiliser array into its normal form. We consider the three cases (a), (b) and (c) in succession.

1. *Case (a)*

This case corresponds to column N_L containing $\mathbb{1}$ only and therefore belongs to the block right of the active region. Step (a.i) does just that and step (a.ii) subsequently excludes this column from the active region.

2. *Case (b)*

This case corresponds to a column containing $\mathbb{1}$ operators and Pauli operators of just one kind. Step (b.i) brings the first of these Pauli operators to the top row, with the ultimate goal of excluding this row from the active region. Step (b.ii) applies a single-qubit rotation to bring the Pauli operators in standard form, which in this case is an X operator.

In step (b.iii) the column is then "cleaned up". Through multiplying the top row K_U with other rows containing an X in column N_L, we obtain a stabiliser array that is still describing the same state but contains only one X in column N_L. So this column is already in standard form.

However, the top row is not in standard form yet. Step (b.iv) applies an appropriate single-qubit operation to every column in the active region, except for the first one, so that the first row contains either $\mathbb{1}$ or X operators. Step (b.v) then performs a "row cleanup", by applying CNOTs to the columns starting with an X, the first column being the control column. The target X operators are thereby turned into $\mathbb{1}$, leaving us with a first row of the form $(X, \mathbb{1}, \ldots, \mathbb{1})$.

It is not a priori clear, however, that step (b.v) is not undoing the cleanup of column N_L by step (b.iii). Nevertheless, inspection of the CNOT truth table reveals that the $\mathbb{1}$ operators in column N_L can either be turned into a Z or remain $\mathbb{1}$, by any number of CNOTs. Although a Z operator can actually occur during the execution of step (b.iii), in the end all operators will be turned back into $\mathbb{1}$. This must be so because the top row of the active region is turned into $(X, \mathbb{1}, \ldots, \mathbb{1})$, which does not commute with a row starting with a Z. So the assumption of commutativity of the generators ensures that step (b.iii) is not undone by step (b.v).

Finally, we note that both the first row and the first column are now in standard form and can be removed from the active region, which is done in step (b.vi). The top left block in the normal form array hereby receives one further X operator.

3. *Case (c)*

The most difficult case to investigate is case (c), because here it is not a priori clear that any progress is made within an iteration. Indeed, the extent of the active region is not changed and it is not clear that further iterations will eventually escape from case (c), resulting potentially in an infinite loop.

However, every execution of case (c) does result in measurable progress. As can be seen from the truth table of the CNOT operation, the end result of the CNOT in step (c.vi) is

	N_L	l	N'_L	l'
K_U	X	X	X	$\mathbb{1}$
$K_U + 1$	Z	Y	$\mathbb{1}$	Y

Hence, a $\mathbb{1}$ is introduced in row K_U where there originally was none. Furthermore, no further algorithmic step in case (c) ever touches this element again, by the following reasoning.

- The only operations that do change the top row K_U are the transposition in step (c.i), and the CNOT in step (c.vi).

- Step (c.i) is executed at most once before the algorithm breaks out of the (c) case, namely at the very beginning. This is because the X brought in the top left position is not changed by the CNOT.

- The first CNOT that operates on target column l introduces the $\mathbb{1}$ there. In further iterations, the CNOT will not operate on column l a second time, because step (c.iv) sets the target column to a column containing an anticommuting pair in the top two rows, and the $\mathbb{1}$ created in the top of column l does not form part of an anticommuting pair.

It is now easy to see why the algorithm must eventually break out of the (c) case. Every iteration through this case increases the number of $\mathbb{1}$ operators in the top row by 1, but there are only a limited number of places (columns) available to do this. Hence the number of successive iterations through case (c) must be limited too.

The algorithm breaks out of the loop through the (c) case when there are no further anticommuting pairs in column N_L. As a consequence, the algorithm will then either execute case (a) or case (b), thereby again reducing the extent of the active region.

C. Alternative proof of projection formulas (3) and (5)

In this subsection we present a new proof of the equivalence of the expressions (2) and (3) for a pure stabiliser state, and of (4) and (5) for a mixed stabiliser state.

By the proof of the CNF1 algorithm, a state described by a certain stabiliser array is unitarily equivalent to the state described by the normal form of that array. Let the initial stabiliser group S be given by a stabiliser array. Let S' be the stabiliser group described by the normal form of that array. The K generators of S' are of the form

$$g'_k = \mathbb{1} \otimes \ldots \otimes X \otimes \ldots \otimes \mathbb{1},$$

with the X operator in the k-th tensor factor. The stabiliser state corresponding to the normal form is therefore

$$\begin{aligned}
\rho' &= \frac{1}{2^N} \sum_{i_1,\ldots,i_K \in \{0,1\}} X^{i_1} \otimes \ldots \otimes X^{i_K} \otimes \mathbb{1}^{\otimes N-K} \\
&= ((\mathbb{1}+X)/2)^{\otimes K} \otimes (\mathbb{1}/2)^{\otimes N-K} \\
&= \frac{1}{2^{N-K}} \prod_{k=1}^{K} \frac{\mathbb{1}+g'_k}{2}.
\end{aligned}$$

Let U be the unitary corresponding to the sequence of elementary operations that brought the stabiliser array to its normal form. To wit, S consists of the elements $g = Ug'U^\dagger$, $g' \in S'$, and can be generated by the generators $g_k := Ug'_kU^\dagger$. Then the stabiliser state corresponding to S is given by

$$\begin{aligned}
\rho &= \frac{1}{2^N} \sum_{g \in S} g \\
&= \frac{1}{2^N} \sum_{g' \in S'} Ug'U^\dagger \\
&= U\rho' U^\dagger \\
&= \frac{1}{2^{N-K}} \prod_{k=1}^{K} U \frac{\mathbb{1}+g'_k}{2} U^\dagger \\
&= \frac{1}{2^{N-K}} \prod_{k=1}^{K} \frac{\mathbb{1}+g_k}{2}.
\end{aligned}$$

□

VI. FIDELITY BETWEEN STABILISER STATES

The topic of this section is an algorithm to calculate the overlap $F = \mathrm{Tr}[\rho_1\rho_2]$ between two mixed stabiliser states ρ_1 and ρ_2 directly from their $K_1 \times N$ and $K_2 \times N$ stabiliser arrays A_1 and A_2.

While the overlap between two states is certainly an interesting quantity, the Bures distance

$$D(\rho_1,\rho_2) := 2\sqrt{1 - F_u(\rho_1,\rho_2)},$$

where F_u is the Uhlmann fidelity

$$F_u(\rho_1,\rho_2) := \mathrm{Tr}[\sqrt{\sqrt{\rho_1}\rho_2\sqrt{\rho_1}}],$$

is a much more desirable quantity, as it is an actual distance measure and has a much nicer interpretation. It is well-known that for pure states the Uhlmann fidelity between two states is just the square root of their overlap, while for general mixed states there is no such relation. It will turn out that with just a minor modification the algorithm is also able to calculate the Uhlmann fidelity. This allows us to calculate the overlap, the Uhlmann fidelity and Bures distance for stabiliser states in one go.

For the calculation of the overlap (and fidelity) it is imperative to take the generator *phases* into account. We will use the vectors S_1 and S_2 for that purpose. The elementary row operations of multiplication and permutation of stabiliser rows are understood to treat the phase vector as an additional column of the stabiliser array. Furthermore, row multiplication, single-qubit rotation and CNOT operation have to multiply the appropriate generator phase with the phase factor mentioned in their truth tables.

A. Algorithm OVERLAP

1. Construct the $(K_1 + K_2) \times N$ composite array A and the composite vector S of generator phases:

$$A = \begin{pmatrix} A_1 \\ \hline A_2 \end{pmatrix} \quad S = \begin{pmatrix} S_1 \\ \hline S_2 \end{pmatrix}.$$

[24]

2. By applying the CNF1 algorithm to the composite array A (and its vector S of generator phases), with initial active region set to the full A_1 part (excluding A_2!), the A_1 part is brought to CNF form, while automatically applying the same sequence of column operations to the A_2 part. Let R_1 be the number of X operators in this CNF.

3. Set the active region to all the rows of the A_2 part and all the columns for which A_1 contains X operators. That is, $K_U = K_1 + 1$, $K_L = K_1 + K_2$, $N_L = 1$ and $N_R = R$. Set $T = 1$.

4. Count the number of different Pauli operators (X, Y and Z) in the first column (N_L) of the active region, i.e. restricting attention only to rows K_U up to K_L.

5. Three cases can be considered:

(a) *There are no Pauli operators in column N_L.*

i. Do nothing.

(b) *There is only 1 kind of Pauli operator.*
Let k be the first row in the active region where column N_L contains a Pauli operator.

i. Make row k the top row of the active region by transposing, if necessary, row k with row K_U.

ii. Multiply row K_U with all other rows in the active region that have a Pauli operator (necessarily equal to the one on row K_U) in column N_L.

iii. Let P be the element in column N_L on row K_U (the column leader).

- If P is not an X, divide T by 2 and increase K_U by 1.
- If P is an X, multiply row N_L (that is, the row containing an X in column N_L of subarray A_1) to row K_U.

(c) *There are at least 2 different kinds of Pauli operators.*
Let k_1 be the first row in the active region where column N_L contains a Pauli operator, and k_2 be the first row in the active region where column N_L contains a different Pauli operator.

i. Make row k_1 the top row of the active region by transposing, if necessary, row k_1 with row K_U.

ii. Make row k_2 the second row of the active region by transposing, if necessary, row k_2 with row $K_U + 1$.

iii. Multiply every other row in the active region with either row K_U, row $K_U + 1$, both rows or none, depending on the element in column N_L (according to Table IV).

iv. Let P_1 be the element in column N_L on row K_U, and P_2 the one on row $K_U + 1$ (the column leaders). Turn P_2 into an X, as follows:

- If neither P_1 nor P_2 is an X, multiply row K_U to row $K_U + 1$, effectively turning P_2 into an X.
- If P_1 is an X, transpose row K_U with $K_U + 1$.

v. Multiply row N_L (that is, the row containing an X in column N_L of subarray A_1) to row $K_U + 1$.

vi. Divide T by 2, and increase K_U by 1.

6. Increase N_L by 1.

7. If the active region is still non-empty ($N_L \leq N_R$ and $K_U \leq K_R$), continue with step 4.

8. (End Game) Here we calculate a correction factor C for the overlap and the fidelity. Set $C = 1$ as default value. If $K_U \leq K_L$ do the following.

(a) Case $N_L \leq N$: Consider the bottom right block of stabiliser array A_2 consisting of rows K_U to K_L and columns $N_R + 1$ to N. Calculate the rank R_2 of that block using, e.g. the RREF algorithm. From R_2 calculate the correction factor as $C = 2^{N-R_1-R_2}$.

(b) Case $N_L > N$: Let t_k be the generator phase of row k. If at least one of the t_k for $K_U \leq k \leq K_L$ is -1, set $T = 0$.

9. Terminate with return values $F = CT/2^{N-K_1+N-K_2}$ for the overlap and $F_u = C\sqrt{T/2^{N-K_1+N-K_2}}$ for the Uhlmann fidelity.

B. Proof of correctness of algorithm OVERLAP

The overlap $F = \mathrm{Tr}[\rho_1\rho_2]$ can be calculated iteratively by performing the trace as a succession of partial traces over single qubits: $F = \mathrm{Tr}[\mathrm{Tr}_1[\rho_1\rho_2]]$, where Tr_1 denotes the partial trace over the first qubit. What we need to show is that one iteration of steps 4-7 indeed performs this single-qubit partial trace. It will be convenient to express the overlap in terms of the projectors $\mathcal{P}_1$ and $\mathcal{P}_2$, with $\rho_1 = \mathcal{P}_1/2^{N-K_1}$ and $\rho_2 = \mathcal{P}_2/2^{N-K_2}$. Then

$$F = \frac{1}{2^{N-K_1+N-K_2}}\,\mathrm{Tr}[\mathcal{P}_1\mathcal{P}_2].$$

Keeping in mind that we also want to calculate F_u, we will proceed by first calculating $\mathcal{P}_1\mathcal{P}_2\mathcal{P}_1$. The overlap is just the trace of this quantity, which is the same as $\mathrm{Tr}[\mathcal{P}_1\mathcal{P}_2]$ by virtue of $\mathcal{P}_1$ being a projector.

Step 2 of the algorithm applies the same sequence of unitaries to both states, hence the overlap between them does not change (and neither does the Uhlmann fidelity). Let $\mathcal{P}_1$ thus be specified by a CNF stabiliser array, containing $R_1 \leq N$ X-operators:

$$\mathcal{P}_1 = \bigotimes_{i=1}^{R_1} \frac{\mathbb{1} + s_i X}{2} \otimes \mathbb{1}^{\otimes N-R_1}.$$

If all generators in array A_1 are independent, we obviously have $R_1 = K_1$. In the above expression, s_i is the generator phase of the i-th generator of $\mathcal{P}_1$. Likewise, we will denote by t_i the generator phase of the i-th generator of $\mathcal{P}_2$. Furthermore, let $\mathcal{P}_1'$ be the stabiliser projector of the array obtained by deleting row 1 and column 1 from A_1.

In the following, we will calculate $\mathcal{P}_1\mathcal{P}_2\mathcal{P}_1$ and show that it is equal to a certain scalar value T times a tensor product of rank-1 projectors and identity operators. We will proceed in an iterative fashion, by showing that $\mathcal{P}_1\mathcal{P}_2\mathcal{P}_1$ decomposes as a scalar T_1 times either a rank-1 projector or an identity tensored with a product $\mathcal{P}_1'\mathcal{P}_2'\mathcal{P}_1'$ of projectors over qubits 2 to N. To calculate T, we start off with the initial value $T = 1$ and update T by multiplying it with the value of T_1 found at each iteration.

We will assume first that A_1 and A_2 have more than 1 column. The case that they only have 1 column, which is what can happen in the final iteration of the algorithm, will be considered in subsection 4. We will also assume that the first tensor factor of $\mathcal{P}_1$ is a rank-1 projector (i.e. $R_1 > 0$). The case that $\mathcal{P}_1$ equals the identity (which will again typically happen at the end of the iterations) will also be covered in subsection 4.

Let us now take on the main case, where there are at least two tensor factors to consider, and the first factor of $\mathcal{P}_1$ is $\frac{\mathbb{1}+s_i X}{2}$. Thus we can write $\mathcal{P}_1 = \frac{\mathbb{1}+s_i X}{2} \otimes \mathcal{P}_1'$. As in algorithm CNF1 there are three cases to consider, depending on the number of different Pauli operators contained in the first column of the second array A_2. We will investigate these three possibilities in succession.

It is useful to note that

$$\frac{\mathbb{1}+sX}{2}\,\sigma\,\frac{\mathbb{1}+sX}{2} = \begin{cases} (\mathbb{1}+sX)/2, & \sigma = \mathbb{1} \\ s(\mathbb{1}+sX)/2, & \sigma = X \\ 0, & \sigma = Y, Z \end{cases}$$

1. Case (a)

If the first column of A_2 contains no Pauli operators, this corresponds to $\mathcal{P}_2$ being of the form

$$\mathcal{P}_2 = \mathbb{1} \otimes \mathcal{P}_2',$$

where $\mathcal{P}_2'$ is the stabiliser projector of the array obtained by deleting column 1 from A_2. Hence

$$\begin{aligned} \mathcal{P}_1\mathcal{P}_2\mathcal{P}_1 &= (\frac{\mathbb{1}+s_1X}{2} \otimes \mathcal{P}_1')\,(\mathbb{1}\otimes\mathcal{P}_2')\,(\frac{\mathbb{1}+s_1X}{2} \otimes \mathcal{P}_1') \\ &= \frac{\mathbb{1}+s_1X}{2} \otimes \mathcal{P}_1'\,\mathcal{P}_2'\,\mathcal{P}_1'. \end{aligned}$$

This is indeed of the form claimed above, with scalar value $T_1 = 1$. Hence, nothing needs to be done in this iteration except for deleting column 1.

2. Case (b)

Steps (b.i) and (b.ii) bring column 1 of $\mathcal{P}_2$ to RREF form. In this case, column 1 will contain a single Pauli operator, σ, in row 1. Denote the remaining operators on row 1 by g'. Let $\mathcal{P}_2'$ be the stabiliser projector of the array obtained by deleting row 1 and column 1 from A_2. Thus $\mathcal{P}_2$ is of the form

$$\mathcal{P}_2 = \frac{\mathbb{1}+t_1\sigma\otimes g'}{2}(\mathbb{1}\otimes\mathcal{P}_2').$$

We then have

$$\begin{aligned} \mathcal{P}_1\,\mathcal{P}_2\,\mathcal{P}_1 &= (\frac{\mathbb{1}+s_1X}{2} \otimes \mathcal{P}_1')\,\frac{\mathbb{1}+t_1\sigma\otimes g'}{2} \\ &\quad \times (\mathbb{1}\otimes\mathcal{P}_2')\,(\frac{\mathbb{1}+s_1X}{2} \otimes \mathcal{P}_1') \\ &= \frac{1}{2}\frac{\mathbb{1}+s_1X}{2} \otimes \mathcal{P}_1'\mathcal{P}_2'\mathcal{P}_1' \\ &\quad + \frac{t_1}{2}\,\frac{\mathbb{1}+s_1X}{2}\,\sigma\,\frac{\mathbb{1}+s_1X}{2} \otimes \mathcal{P}_1'g'\mathcal{P}_2'\mathcal{P}_1'. \end{aligned}$$

We can therefore distinguish two cases. If σ is not an X, we find

$$\mathcal{P}_1\,\mathcal{P}_2\,\mathcal{P}_1 = \frac{1}{2}\frac{\mathbb{1}+s_1X}{2} \otimes \mathcal{P}_1'\mathcal{P}_2'\mathcal{P}_1'.$$

This corresponds to a value of $T_1 = 1/2$. This is implemented in step (b.iii, first case) by dividing the running T by 2, and deleting row 1 and column 1 from A_2.

If, on the other hand, $\sigma = X$, we have

$$\begin{aligned} \mathcal{P}_1\,\mathcal{P}_2\,\mathcal{P}_1 &= \frac{1}{2}\frac{\mathbb{1}+s_1X}{2} \otimes \mathcal{P}_1'\mathcal{P}_2'\mathcal{P}_1' \\ &\quad + \frac{s_1t_1}{2}\,\frac{\mathbb{1}+s_1X}{2} \otimes \mathcal{P}_1'g'\mathcal{P}_2'\mathcal{P}_1' \\ &= \frac{\mathbb{1}+s_1X}{2} \otimes \mathcal{P}_1'\,\frac{\mathbb{1}+s_1t_1g'}{2}\,\mathcal{P}_2'\,\mathcal{P}_1' \\ &= \frac{\mathbb{1}+s_1X}{2} \otimes \mathcal{P}_1'\,\mathcal{P}_2''\,\mathcal{P}_1'. \end{aligned}$$

where $\mathcal{P}_2'' = \frac{\mathbb{1}+s_1t_1g'}{2}\,\mathcal{P}_2'$ is a projector. This corresponds to a scalar value of $T_1 = 1$. This is accomplished in step (b.iii, second case) by multiplying row 1 of A_1 to row 1 of A_2, and deleting column 1 of A_2 (leaving row 1).

3. Case (c)

Steps (c.i), (c.ii) and (c.iii) bring column 1 of A_2 in RREF form. Column 1 will contain two Pauli operators, σ_1 in row 1, and $\sigma_2 \neq \sigma_1$ in row 2. Step (c.iv) ensures, by suitable row multiplication or transposition, that σ_2 is an X operator, so σ_1 is not. Let the remaining operators on rows 1 and 2 be denoted by g_1' and g_2', respectively. Let $\mathcal{P}_2'$ be the stabiliser projector of the array A_2', obtained by deleting rows 1 and 2 and column 1 from A_2. Then $\mathcal{P}_2$ is given by

$$\mathcal{P}_2 = \frac{\mathbb{1}+t_1\sigma_1\otimes g_1'}{2}\,\frac{\mathbb{1}+t_2X\otimes g_2'}{2}\,(\mathbb{1}\otimes\mathcal{P}_2').$$

Thus

$$\begin{aligned} &\mathcal{P}_1\,\mathcal{P}_2\,\mathcal{P}_1 \\ &= (\frac{\mathbb{1}+s_1X}{2} \otimes \mathcal{P}_1') \\ &\quad \times \frac{\mathbb{1}+t_1\sigma_1\otimes g_1'}{2}\,\frac{\mathbb{1}+t_2X\otimes g_2'}{2} \\ &\quad \times (\mathbb{1}\otimes\mathcal{P}_2')\,(\frac{\mathbb{1}+s_1X}{2} \otimes \mathcal{P}_1') \\ &= \frac{1}{4}\Big[\frac{\mathbb{1}+s_1X}{2} \otimes \mathcal{P}_1'\mathcal{P}_2'\mathcal{P}_1' \\ &\quad + t_1\frac{\mathbb{1}+s_1X}{2}\sigma_1\frac{\mathbb{1}+s_1X}{2} \otimes \mathcal{P}_1'g_1'\mathcal{P}_2'\mathcal{P}_1' \\ &\quad + t_2\frac{\mathbb{1}+s_1X}{2}X\frac{\mathbb{1}+s_1X}{2} \otimes \mathcal{P}_1'g_2'\mathcal{P}_2'\mathcal{P}_1' \\ &\quad + t_1t_2\frac{\mathbb{1}+s_1X}{2}\sigma_1X\frac{\mathbb{1}+s_1X}{2} \otimes \mathcal{P}_1'g_1'g_2'\mathcal{P}_2'\mathcal{P}_1'\Big] \\ &= \frac{1}{4}\Big[\frac{\mathbb{1}+s_1X}{2} \otimes \mathcal{P}_1'\mathcal{P}_2'\mathcal{P}_1' \\ &\quad + s_1t_2\frac{\mathbb{1}+s_1X}{2} \otimes \mathcal{P}_1'g_2'\mathcal{P}_2'\mathcal{P}_1'\Big], \end{aligned}$$

giving

$$\begin{aligned}\mathcal{P}_1\,\mathcal{P}_2\,\mathcal{P}_1 &= \frac{1}{2}\frac{\mathbb{1}+s_1X}{2}\otimes\mathcal{P}_1'\,\frac{\mathbb{1}+s_1t_2g_2'}{2}\mathcal{P}_2'\mathcal{P}_1' \\ &= \frac{1}{2}\frac{\mathbb{1}+s_1X}{2}\otimes\mathcal{P}_1'\mathcal{P}_2''\mathcal{P}_1'\end{aligned}$$

with $\mathcal{P}_2''=\frac{\mathbb{1}+s_1t_2g_2'}{2}\mathcal{P}_2'$ a projector. This corresponds to $T_1 = 1/2$. This is implemented in steps (c.v) and (c.vi) through multiplying row 2 in A_2 by row 1 of A_1, dividing T by 2, and subsequently deleting row 1 and column 1 in A_2.

4. *End Game*

We still have to consider the situation where there is only one column left and the one where $\mathcal{P}_1$ is a tensor product of identity operators.

The first situation is when $N_L = N$. In that case the symbols $\mathcal{P}_1'$, g' and $\mathcal{P}_2'$ used in the previous subsections are meaningless. However, we can still make sense out of the calculations if we replace these symbols formally by the scalar 1. Inspection of the relevant calculations then shows that at the very end of the algorithm, if $N_L = N$ we have to check whether one of the remaining generator phases t_i is -1, in which case the states under consideration are orthogonal. That means both the overlap and the Uhlmann fidelity are 0, which we impose by setting $T = 0$.

If $N_L < N$ but $\mathcal{P}_1$ acts as the identity on columns N_L to N, $\mathcal{P}_1\mathcal{P}_2\mathcal{P}_1$ reduces to $\mathcal{P}_2$. This can easily be decomposed as a tensor product by calculating its rank R_2 (the easiest way to do this is by using the RREF algorithm). Thus the remaining part on columns N_L to N of $\mathcal{P}_1\mathcal{P}_2\mathcal{P}_1 = \mathcal{P}_2$ is unitarily equivalent to

$$\bigotimes_{i=1}^{R_2}\frac{\mathbb{1}+X}{2}\otimes\mathbb{1}^{\otimes N-R_1-R_2}.$$

5. *Overlap and Uhlmann Fidelity*

In the previous subsection we have shown that $\mathcal{P}_1\mathcal{P}_2\mathcal{P}_1$ is equal to T times a tensor product of R_1+R_2 rank-1 projectors and $N-R_1-R_2$ identity operators. Calculating the overlap and the Uhlmann fidelity is now easy. Assuming that $R_1 = K_1$, we have for the overlap

$$\begin{aligned}F &= \frac{1}{2^{N-K_1+N-K_2}}\,\mathrm{Tr}[\mathcal{P}_1\mathcal{P}_2\mathcal{P}_1] \\ &= \frac{1}{2^{N-K_1+N-K_2}}\,T\,2^{N-R_1-R_2} \\ &= 2^{-(N-K_2+R_2)}\,T,\end{aligned} \tag{6}$$

where it has to be noted that $R_2 \leq K_2$. Since T is also a negative power of 2 one sees that the overlap takes values of either 0 or 2^{-j}, where j is an integer between 0 and N.

Similarly, the Uhlmann fidelity between states ρ_1 and ρ_2 is given by

$$F_u = \mathrm{Tr}[\sqrt{\sqrt{\rho_1}\rho_2\sqrt{\rho_1}}].$$

Again we substitute the stabiliser states for their appropriately scaled projectors. Noting that the square root of a projector is that same projector gives

$$\begin{aligned}F_u &= (2^{N-K_1}\,2^{N-K_2})^{-1/2}\,\mathrm{Tr}[\sqrt{\mathcal{P}_1\mathcal{P}_2\mathcal{P}_1}] \\ &= (2^{N-K_1}\,2^{N-K_2})^{-1/2}\,\sqrt{T}\,2^{N-R_1-R_2} \\ &= 2^{(K_2-K_1)/2-R_2}\,\sqrt{T}.\end{aligned} \tag{7}$$

VII. BIPARTITE NORMAL FORM

In this Section, we will modify the single-party algorithm CNF1 so that it can be used to reduce a stabiliser array of a bipartite system to a certain normal form. This algorithm will allow us to deduce the exact structure of this normal form, which is the content of Theorem 1. This Theorem basically tells us that a bipartite mixed stabiliser state is locally equivalent to a tensor product of some number of pure EPR pairs and a separable mixed state. The main benefit of this normal form is that the entanglement of the state can immediately be read off from the normal form. Because of the Theorem, it turns out that in order to calculate the state's entanglement it is not necessary to actually compute the normal form completely. Instead, a simplified algorithm to calculate entanglement will be presented.

Let us start with the statement of the normal form.

Theorem 1 *Consider a system of N qubits, separated into two parties, A and B, containing N_A and N_B qubits, respectively. Consider a stabiliser state described by an array of K independent, commuting generators.*

i) By applying a suitable sequence of elementary row operations and local elementary column (qubit) operations, the stabiliser array can be brought into the following normal form:

$$\left(\begin{array}{cccc|ccc||cccc|ccc}
X & \mathbb{1} & \ldots & \mathbb{1} & \mathbb{1} & \ldots & \mathbb{1} & X & \mathbb{1} & \ldots & \mathbb{1} & \mathbb{1} & \ldots & \mathbb{1} \\
Z & \mathbb{1} & \ldots & \mathbb{1} & \mathbb{1} & \ldots & \mathbb{1} & Z & \mathbb{1} & \ldots & \mathbb{1} & \mathbb{1} & \ldots & \mathbb{1} \\
\mathbb{1} & X & \ldots & \mathbb{1} & \mathbb{1} & \ldots & \mathbb{1} & \mathbb{1} & X & \ldots & \mathbb{1} & \mathbb{1} & \ldots & \mathbb{1} \\
\mathbb{1} & Z & \ldots & \mathbb{1} & \mathbb{1} & \ldots & \mathbb{1} & \mathbb{1} & Z & \ldots & \mathbb{1} & \mathbb{1} & \ldots & \mathbb{1} \\
\vdots & \vdots & & \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots & \vdots & & \vdots \\
\mathbb{1} & \mathbb{1} & \ldots & X & \mathbb{1} & \ldots & \mathbb{1} & \mathbb{1} & \mathbb{1} & \ldots & X & \mathbb{1} & \ldots & \mathbb{1} \\
\mathbb{1} & \mathbb{1} & \ldots & Z & \mathbb{1} & \ldots & \mathbb{1} & \mathbb{1} & \mathbb{1} & \ldots & Z & \mathbb{1} & \ldots & \mathbb{1} \\
\hline
\mathbb{1} & \mathbb{1} & \ldots & \mathbb{1} & * & \ldots & * & \mathbb{1} & \mathbb{1} & \ldots & \mathbb{1} & * & \ldots & * \\
\vdots & \vdots & & \vdots & * & \ldots & * & \vdots & \vdots & & \vdots & * & \ldots & * \\
\mathbb{1} & \mathbb{1} & \ldots & \mathbb{1} & * & \ldots & * & \mathbb{1} & \mathbb{1} & \ldots & \mathbb{1} & * & \ldots & *
\end{array}\right) \tag{8}$$

Here, the asterisk stands for either $\mathbb{1}$ or X, and the double line is the separation line between the two parties.

ii) Every pair of rows containing the XZ combinations corresponds to two qubits (one from each party) being in a pure *maximally entangled EPR state and completely disentangled from the other qubits. The rows in the lower blocks of the normal form, containing only $\mathbb{1}$ and X operators, correspond to the remaining qubits being in a (in general, mixed) separable state.*

iii) The stabiliser state described by the stabiliser array is locally equivalent to a tensor product of a certain number p of EPR pairs Ψ with a separable state. For any additive entanglement measure E, the entanglement of the stabiliser state is $pE(\Psi)$. An upper bound on p is given by

$$p \leq \min(\lfloor K/2 \rfloor, N_A, N_B). \tag{9}$$

Equality is obtained if and only if $K = 2N_A = 2N_B$.

We will postpone the proof of part i) of the Theorem, the normal form, to the end of this Section. The proof of part iii), the entanglement properties of the normal form, is elementary and is left to the reader. The proof of part ii) is presented next.

Proof of part ii). For convenience of notation, we first permute the qubits in such a way that the pairs of columns having an XZ pair in the same rows are adjacent. By eq. (5), the stabiliser state corresponding to the normal form of the Theorem is

$$\rho = \frac{1}{2^{N-K}} \prod_{k=1}^{2p} \frac{\mathbb{1} + g_k}{2} \prod_{l=2p+1}^{K} \frac{\mathbb{1} + g_l}{2}.$$

From the specific form of the generators one sees that ρ can be written as tensor product

$$\rho = \left(\frac{\mathbb{1} + X \otimes X}{2} \, \frac{\mathbb{1} + Z \otimes Z}{2} \right)^{\otimes p} \otimes \rho',$$

where ρ' corresponds to the factor $\prod_{l=2p+1}^{K} (\mathbb{1} + g_l)/2$ containing $\mathbb{1}$ and X operators only. It is a simple matter to verify that the factor $(\mathbb{1} + X \otimes X)(\mathbb{1} + Z \otimes Z)/4$ is identical to the EPR state $\Psi = |\psi\rangle\langle\psi|$, with $\psi = (1, 0, 0, 1)^T/\sqrt{2}$.

It is also simple to see that ρ' is a separable state. As it only contains $\mathbb{1}$ and X operators, it is diagonal in any basis where X is diagonal, and it is well-known and easy to see that diagonal states are separable. □

A. Algorithm CNFP for calculating the entanglement

We now present an algorithm to calculate the number of EPR pairs in the normal form, without actually reducing the stabiliser array completely to that normal form. This algorithm is almost identical to algorithm CNF1, the reduction algorithm for the single-party case.

To calculate the entanglement, the initial active region is set to comprise the block of elements pertaining to party A only, rather than the full stabiliser array, and algorithm CNFP (CNF for a single Party) is run on this active region.

Algorithm CNFP is identical to CNF1, apart from the following two differences:

Step (b.vi). While in CNF1 step (b.iii) is never undone by step (b.v) due to commutativity of the generators, this need no longer be the case here. Indeed, here we restrict attention to one party only, and the parts of the generators local to party A need not commute. Hence step (b.v) might leave Z operators in the leftmost column of the active region. We thus need a modification here: first we must check whether this has happened and only if there are no Z operators in this column may K_U and N_L be increased by 1. Otherwise, the extent of the active region must stay the same. The additional Z's will then be treated in the next iteration of the algorithm.

Initial stabiliser array:

$$\begin{pmatrix} X & . \\ Z & . \end{pmatrix}$$

Depending on the content of column 2, do the following:

$\mathbb{1}X$:

$\mathbb{1}Y$:

$\mathbb{1}Z$: Using a single-qubit operation, bring column 2 to $\mathbb{1}Z$, then perform a CNOT with column 1 as target (!) and column 2 as control.

$X\mathbb{1}$:

$Y\mathbb{1}$:

$Z\mathbb{1}$: Using a single-qubit operation, bring column 2 to $X\mathbb{1}$, then perform a CNOT with column 1 as control and column 2 as target.

XX:

YY:

ZZ: Using a single-qubit operation, bring column 2 to ZZ, then perform a CNOT with column 1 as target (!) and column 2 as control. Column 2 now contains $Z\mathbb{1}$. Apply another single-qubit operation to bring this to $X\mathbb{1}$, and (as in the above cases) perform a CNOT with column 1 as control and column 2 as target.

TABLE V: Required operations to eliminate all Pauli operators from column 2 of the stabiliser array shown above, in the various cases encountered.

Step (c.iv). In step (c.iv), the original algorithm looked for an anticommuting pair in the top two rows, the presence of which having been guaranteed by generator commutativity. Here, again, this is no longer true, because the pair might be located in party B, which we are not allowed to touch here. We therefore need a second modification, to deal with the case that there is no such anticommuting pair. In that case, instead of steps (c.v), (c.vi) and (c.vii), the following operations must be executed. Recall that the first column has an XZ pair in its first two rows. This pair can now be used to eliminate all other Pauli operators in both the first two rows (by suitable single-qubit operations and CNOTs) and in the first column (by suitable row multiplications). Tables V and VI contain the details. After that, the XZ pair can be split off from the active region to form part of the normal form, by increasing K_U by 2, and N_L by 1.

Algorithm CNFP brings only that part of the stabiliser array

Initial stabiliser array:

$$\begin{pmatrix} X \\ Z \\ . \end{pmatrix}$$

Depending on the content of row 3, do the following:

X: Multiply row 1 with row 3.

Y: Multiply row 1 with row 3, and then row 2 with row 3.

Z: Multiply row 2 with row 3.

TABLE VI: Required operations to eliminate any Pauli operator from row 3 of the stabiliser array shown above.

in normal form that belongs to party A. Nevertheless, this is enough to read off the number of EPR pairs in the full reduction. This will be proven below. The number of EPR pairs p is simply given by the number of XZ pairs in the normal form of party A.

B. Proof of part i)

By suitable modification of the Proof of algorithm CNF1, it can be shown that algorithm CNFP brings that part of the stabiliser array belonging to party A to the form as shown in (8), the columns left of the double vertical line.

We next show that by further applying suitable column operations on the columns of party B, the complete normal form of (8) can be obtained.

Consider the first XZ pair in party A. By commutativity of the generators, there must at least be 1 anticommuting pair on the same rows in party B. By a column permutation and a suitable single-qubit rotation, this anticommuting pair can be moved to the first column of party B and be brought in XZ form. Using suitable CNOTs (see Table V) the operators right of the XZ pair can all be brought to an $\mathbb{1}$ operator. Again by commutativity, the operators below the XZ pair must then automatically be all $\mathbb{1}$ operators. Indeed, if a row (below the second) contained a Pauli operator in the first column of party B, it would not commute with either the first row, the second, or both.

One can proceed in a similar fashion with the second of party A's XZ pairs and party B's second column and third and fourth row, and so forth until all of A's XZ pairs have been treated in this way.

What remains then are the rows below the horizontal line in (8). To show that the lower right block of party B in (8) can be brought to the form as advertised (i.e. containing only X and $\mathbb{1}$ operators, as denoted by the asterisks), we note that party A contains no anticommuting pairs in those rows. Hence, the subarray consisting of party B's lower right block (restricted to that block's columns) consists of mutually commuting generators. By applying algorithm CNF1 to that subarray it can be brought in single-party normal form, consisting of X and $\mathbb{1}$ operators only. Evidently, the row operations performed by the CNF1 algorithm (row permutation and multiplication) will also affect the corresponding rows in party A. However, as party A has only X and $\mathbb{1}$ operators in those rows, no Y or Z operators will be introduced, and the end result will also contain only X and $\mathbb{1}$ operators. □

VIII. CONCLUSION

The stabiliser formalism is a convenient tool for the study of entanglement properties of large quantum many-body systems. While the stabiliser formalism provides an efficient description of the quantum state in terms of eigenvalue equations, it is not immediately obvious how to obtain physical properties directly from these eigenvalue equations without explicitly having to write out the corresponding quantum state. In this paper we have presented, employing elementary tools, a number of normal forms for pure *and* mixed stabiliser states. We have furthermore provided explicit, detailed descriptions of algorithms, whose convergence we have proven, that allow the generation of these normal forms. Using these normal forms, we can compute *any* entanglement measure, overlaps between stabiliser states and various other quantities. Detailed descriptions of the algorithms are provided that should make it straightforward to implement them in any programming language and we are able to provide MatLab suite of programs on request.

These algorithms provide a firm basis for the exploration entanglement properties of stabiliser states and suitable generalisations in a great variety of contexts. For example, it is readily seen that our approach is suitable for the efficient simulation of systems where the initial state is a linear combination of a polynomial number of stabiliser states. This and other applications will be explored in forthcoming publications.

Acknowledgments

We would like to thank H.J. Briegel, O. Dahlsten and J. Oppenheim for discussions. This work is part of the QIP-IRC (www.qipirc.org) supported by EPSRC (GR/S82176/0) and is also supported by the EU Thematic Network QUPRODIS (IST-2001-38877) and the Leverhulme Trust grant F/07 058/U. Finally, we thank the anonymous referees for comments that substantially improved the presentation, and also for pointing out a serious omission in our first treatment of overlap and Uhlmann fidelity.

[1] M.B. Plenio and V. Vedral, Cont. Phys. **39**, 431 (1998).

[2] J. Eisert and M.B. Plenio, Int. J. Quant. Inf. **1**, 479 (2003).

[3] M.B. Plenio and S. Virmani, E-print arXiv quant-ph/0504163.

[4] N. Linden, S. Popescu, B. Schumacher and M.Westmoreland, E-print arXiv quant-ph/9912039; E.F. Galvão, M.B. Plenio and S. Virmani, J. Phys. A **33**, 8809 (2000); A. Acin, G. Vidal and J.I. Cirac, Phys. Rev. A **62**, 062314 (2000); S. Ishizaka and M.B. Plenio, Phys. Rev. A **71**, 052303 (2005).

[5] D. Gottesman, Stabilizer Codes and Quantum Error Correction, PhD Thesis, CalTech 1997, E-print arXiv quant-ph/9705052.

[6] Daniel Gottesman: The Heisenberg Representation of Quantum Computers, E-print arXiv: quant-ph/9807006.

[7] H.-J. Briegel and R. Raussendorf, Phys. Rev. Lett. **86**, 910 (2001).

[8] M.A. Nielsen and I.L Chuang, *Quantum computation and quantum information*, Cambridge University Press, 2000.

[9] E. Hostens, J. Dehaene, and B. De Moor, Stabilizer states and Clifford operations for systems of arbitrary dimensions, and modular arithmetic, Phys. Rev. A **71**, 042315 (2005).

[10] M. Van den Nest, J. Dehaene, and B. De Moor, On local unitary versus local Clifford equivalence of stabilizer states, E-print arXiv quant-ph/0411115.

[11] D. Schlingemann, Quant. Inf. Comp. **2**, 307 (2002).

[12] D. Schlingemann, Quant. Inf. Comp. **4**, 289 (2004).

[13] M. Hein, J. Eisert, and H.-J. Briegel, Phys. Rev. A **69**, 062311 (2004).

[14] M. Van den Nest, J. Dehaene, and B. De Moor, Graphical description of the action of local Clifford transformations on graph states, Phys. Rev. A **69**, 022316 (2004).

[15] O. Guehne, G. Toth, Ph. Hyllus and H.-J. Briegel, Bell Inequalities for Graph States, E-print arXiv quant-ph/0410059.

[16] W. Dür, H. Aschauer and H.-J. Briegel, Multiparticle entanglement purification for graph states, Phys. Rev. Lett. **91**, 107903 (2003).

[17] M. Van den Nest, J. Dehaene, and B. De Moor, An efficient algorithm to recognize local Clifford equivalence of graph states, Phys. Rev. A **70**, 034302 (2004)

[18] M. Van den Nest, J. Dehaene, and B. De Moor, The invariants of the local Clifford group, Phys. Rev. A **71**, 022310 (2005).

[19] S. Anders and H.-J. Briegel, E-print arXiv quant-ph/0504117.

[20] J. Calsamiglia, L. Hartmann, W. Dür, and H.-J. Briegel, E-print arXiv quant-ph/0502017

[21] S. Aaronson and D. Gottesman, E-print arXiv quant-ph/0406196.

[22] D. Schlingemann, Quant. Inf. Comp. **3**, 431 (2003).

[23] After completion of the scientific work reported in this paper, we have become aware of D. Fattal, T. S. Cubitt, Y. Yamamoto, S. Bravyi and I. L. Chuang, E-print arXiv quant-ph/0406168 (2004).

[24] This composite array is no longer a stabiliser array because generators from A_1 need not commute with those of A_2. Even worse, A_1 and A_2 may generate conflicting stabilisers, i.e. with opposite phase factors. One thus cannot just apply reduction algorithms, say RREF, to A as a whole, and claim to have calculated the rank of A; in fact, the rank of A is just not defined. There are, however, some situations where one can treat the composite array, with due care, as if it were a stabiliser array. This is especially useful when one needs to apply the same column operations to A_1 and A_2.